Automatic Embedding of a Ferroelectric Capacitor Inside the Circuit Simulator Eldo

M. Zorzi, N. Speciale, G. Masetti

DEIS, University of Bologna
Viale Risorgimento 2 40136 Bologna, Italy

Abstract

In this paper we describe the implementation of a ferroelectric capacitor model inside a commercial circuit simulator by using I.M.A.Ge.*, a new versatile CAD tool to automatic embed semiconductor device models. By using* I.M.A.Ge.*, user defined models can be developed quickly, allowing efficient simulation of even large circuits. To outline the performance improvement and flexibility obtained with our tool, we present comparisons with both simulation results of large memory array and simulation of HDL-A ferroelectric capacitor implementation.*

Keywords: Semiconductor, CAD tool, Circuit simulation, Device Modeling, Behavioral Modeling, Ferroelectric Capacitor.

1 Introduction

The growth of new advanced technologies and the need to have updated models during circuit simulation, have induced the development of new CAD tools that allow the implementation of new devices inside the commercial simulators. Unfortunately, it is difficult to develop complete, accurate and numerically robust models that work properly and easily integrate them within standard simulators and CAD tools in an industrial IC design environment.

Despite of the most recent developments [1][3], the proposed *I.M.A.Ge.* high-level language is mainly devoted to describe low-level device model. It allows access and control to all the key quantities during each newton and transient iteration steps, thus avoiding convergence problems that are hard to control in other high-level languages. As an example, we will show the implementation of a ferroelectric capacitor [11], a novel device which can provide an alternative to the already affirmed floating-gate memories. Moreover, since the model is compiled as a dynamic loaded library, very high performances can be obtained during circuit simulation and a great number of devices can be simulated without numerical problems.

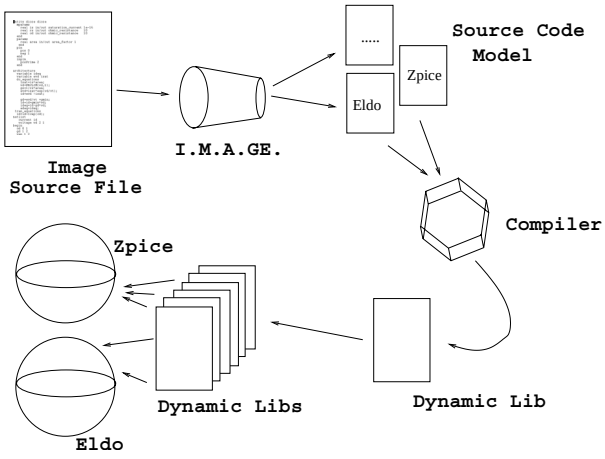

Figure 1: Interaction between *I.M.A.Ge.* and different circuit simulators.

The paper is organized as follows: in the next Section we introduce the *I.M.A.Ge.* tool, showing its syntax and explaining how a model is built. Section 3 describes the algorithm used to implement the ferroelectric capacitor, while Section 4 reports simulation results and comparisons with a full HDL-A implementation. Finally, conclusions will be drawn in Section 5.

2 Overview of the *I.M.A.Ge.* tool

The program *I.M.A.Ge.* (**I**nternal **M**odel **A**utomatic **GE**nerator) represents an efficient solution to cope with the problem of automatic code model generation and implementation. Presently, the proposed tool can automatically generate portable code suitable to be used with both different circuit simulators based on the original Berkeley Spice [8] [9] and the commercial simulator Eldo [2] [6].

2.1 Model Definition

Adding a new model in a simulator is a two-step process: first, *I.M.A.Ge.* takes as input a file written in a simple language, describing the device model. Then, it creates a library that can be loaded inside Eldo or the spice-based simulator developed by our group, called Zpice, which provides a dynamic support to the device models. By so doing, all the device models remain external to the simulator core and are dynamically loaded only at run-time, as shown in figure 1. This solution allows great flexibility and makes the process of incorporating a new model brief, since the core of the simulator is left unmodified and only the effective new adding library has to be compiled.

The *I.M.A.Ge.* source language is quite intuitive since it borrows many syntax constructs from well known behavioral languages such as VHDL and Spice netlist source file [5].

```
entity <model> <modelName>
  mparams
   <model param>
   ...
  end
  params
   <instance param>
   ...
  end
  pin
    <name> <number>
    ...
  end
  inpin
    <name> <number>
    ...
  end
  architecture
   variable <name>
   equations
     <equations>
   netlist
     current|voltage <name>
   begin
     <netlist definition>
   end
```

Figure 2: Example of *I.M.A.Ge.* source file template.

Every model implemented with *I.M.A.Ge.* must be defined by using a *companion-model* which describes currents and voltages behavior mapped on model internal and external nodes. Even if the use of a *companion-model* could seem very difficult, it offers a quite simple method to convert a set of equations in an equivalent circuit. Explicit declaration of derivatives speeds up simulation, especially when a large number of devices are involved, giving a better performance in comparison with methods that adopt numerical derivatives. *I.M.A.Ge.* checks with care basic rules to be obeyed for proper execution such as the model definitions and initial conditions and prevents the user to implement an ill-defined device that can give numerical problems during simulation, so reducing convergence problems.

All this features place *I.M.A.Ge.* at an intermediate level between low-level language near to "pure" C-code and high level languages like VHDL-AMS.

2.2 *I.M.A.Ge.* language syntax

The syntax for a generic model is reported in Figure 2: in the first line the name of the model is defined, then the sections **mparams** and **params** are used to introduce the model and instance parameters; **pin** and **inpin** are used to define respectively the external and internal nodes; section **architecture** defines the equations for the several quantities, while in the **netlist** the circuit for the internal device companion model is reported. Finally, the purpose of the **equations** section is to define the *companion-model* quantities, as the charge in a ferroelectric capacitor implemented by the algorithm described in the following Section.

3 *I.M.A.Ge.* description of the ferroelectric capacitor

The *I.M.A.Ge.* input file of the ferroelectric capacitor is very simple (figure 3), since this device does not have internal nodes. It is composed by a two terminal model with a charge Q_d having the typical hysteresis characteristic that is represented by a set of equations defining saturated (Q_{dsat}) and non saturated (Q_d) loops as follows [10]:

$$\frac{dQ_d(V)}{dV} = \left[1 - \tanh\sqrt{\left(\frac{Q_d - Q_{dsat}}{\xi Q_s - Q_d}\right)}\right]\frac{dQ_{dsat}(V)}{dV}$$

$$Q_{dsat}(V) = \xi\, Q_s \tanh\left[\frac{(\xi V - V_c)}{2\delta}\right]$$

where Q_s is the saturation charge, V_c is the coercive voltage and

$$\xi = sign(\frac{dV}{dt}) \qquad (1)$$

$$\delta \quad = \quad V_c \left[\log \left(\frac{1 + Q_r/Q_s}{1 - Q_r/Q_s} \right) \right]$$

The pseudo-code of the algorithm used to implement the ferroelectric capacitor is shown in figure 4. During the hysteresis characteristic calculation, the storage of some important quantities is accomplished by using state variables, which are never modified by the simulator kernel. We use five states to store *vcap* and *qcap*, the capacitor bias and charge at each newton step, *vcap_t*0 and *qcap_t*0, the capacitor bias and charge calculated on the previous transient step, *dqdv*, the charge derivative with respect to the capacitor bias calculated at each newton step. In figure 4 auxiliary variables are defined as *vcap_prev*, which is the capacitor bias at the previous newton step.

The algorithm is based on the calculation of the capacitor quantities depending on some global flags, which let us to follow the simulation during all steps: *tran_step_done* is a flag set to "1" if a transient step has been solved, *init_tran_step* is a flag set to "1" if the simulator is solving the first transient step. If these flags are set to "1", then we rotate some states, being for example *qcap_t*0 = *qcap*, where *qcap* is the capacitor charge calculated on previous newton step.

When the simulator has solved the transient step at time $t0$, the capacitor charge Q_d must be evaluated from the value of *qcap_t*0, using derivative *dqdv* calculated during the previous newton step and limiting the bias variation on capacitor since excessive fluctuations can be induced. Moreover, between each transient step the simulator can perform many newton iterations, so capacitor charge at iteration n_i is calculated from the previous iteration n_{i-1}, according to the following formula:

$$Q(n_i) \quad = \quad Q(n_{i-1}) + \left. \frac{dQ}{dV} \right|_{n_{i-1}} \cdot \Delta V \qquad (2)$$

where ΔV is the bias variation up limited by a quantity called *max_delta_v*.

A more accurate approach is to calculate $Q_d(n_i)$ from $Q_d(n_{i-1})$ using both first and second order derivatives; using this solution, *max_delta_v* could be chosen greater than previous, leading to a small number of newton iterations with the same numerical error.

Finally, particular attention must be paid to the evaluation of ξ in equation (1), that is calculated from *vcap_t*0 and *vcap*. In fact, numerical errors can introduce discontinuities in Q_d shape, breaking the numerical robustness of the implemented model.

```
entity ferrocap Ferrocap1
  mparams
    real Qs in/out . . .
    real Qr in/out . . .
    real Vc in/out . . .
    real Ic in/out . . .
  end
. .
architecture
  variable vcap, xi
  variable delta_v
  variable dqdv
  . .
end
```

Figure 3: *I.M.A.Ge.* implementation of the ferroelectric capacitor.

```
if (init_tran_step) or (tran_step_done) then
   state0=state1;
   state2=state3;

delta_v = vcap−vcap_prev;
q_n0    = state3;

if (delta_v >= 0) then
   delta_v_sign = +1;
else
   delta_v_sign = −1;

if ((vcap−state0)>=0) then
   xi = 1;
else
   xi = −1;

if (abs(delta_v)>max_delta_v) then
   delta_v = delta_v_sign * max_delta_v;
   vcap    = vcap_prev + delta_v;
dqdv = state4;
delta_q = delta_v*dqdv;
q_n1 = q_n0 + delta_q;

dqdv = calculate_new_dqdv(q_n1);
state1 = vcap;
state3 = q_n1;
state4 = dqdv;
```

Figure 4: Pseudo-code of the ferroelectric capacitor model.

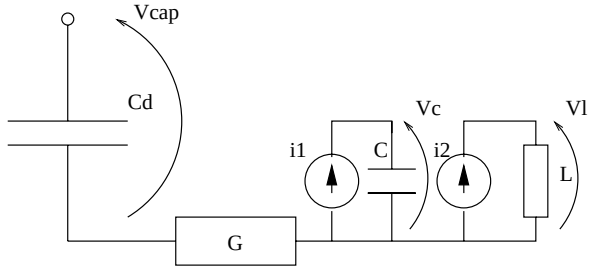

Figure 5: Ferroelectric capacitor *companion-model* derived from equation (3).

Another possible approach to describe the ferroelectric model in *I.M.A.Ge.* is to modify equation (1) in order to obtain

$$\frac{dQ_d(V)}{dt} = F(Q_d, V)\frac{dV}{dt} \qquad (3)$$

which can be implemented as the companion model showed in figure 5, where

$$\begin{aligned} i_1 &= \frac{dQ_d(V_{cap})}{dt} \\ i_2 &= V_{cap} \\ Q_d &= V_c/C \\ \frac{dV_{cap}}{dt} &= V_l/L \end{aligned}$$

4 Simulation Results

The QV characteristic capacitor charge is showed in figure 6.

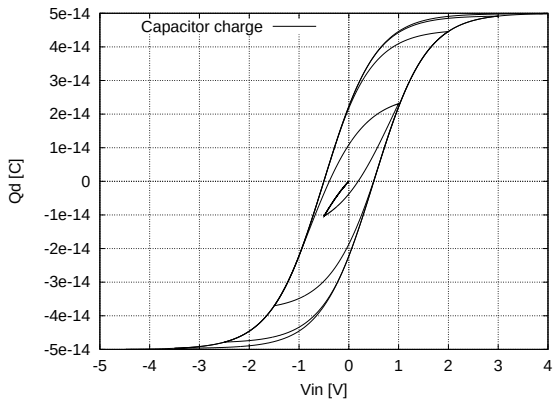

Figure 6: Simulated saturated charge hysteresis loop including minor internal loops.

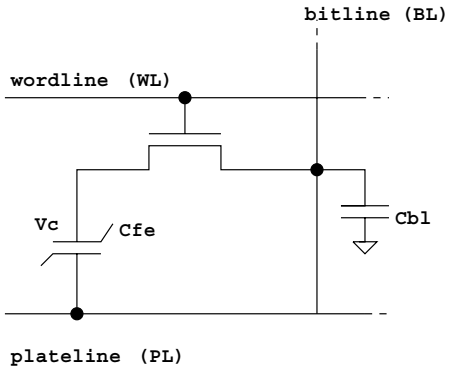

Figure 7: Ferroelectric 1T-1C memory cell. "Cbl" represents the total parasitic capacitance of the bitline and "Cfe" the ferroelectric capacitor.

The memory cell simulated is based on the 1T-1C schema [11]: the cell, as shown in figure 7, consists of a single ferroelectric capacitor C_{fe} that is connected to a plate line (PL) at one end and via an access transistor to a bit line (BL) at the other end. The cell is accessed by raising the word line (WL) and hence turning on the access transistor. We use a simple Level 3 model for the MOS transistor, and the whole circuit was composed by an array of 64k ferroelectric cells with WL/PL architecture.

The purpose of the simulation was to test the numerical robustness of the implemented model by simulating various successive write and read cycles on memory words, according to the timing diagram showed in figure 8. To write a "1", the BL is raised to V_{dd}, then the transistor is turned on by raising WL. At this moment the charge in the ferroelectric is independent of its initial state, and at the end of the cycle the capacitor charge is a negative charge state. To write a "0" the BL is driven to 0 V before the activation of the WL, and at the end of the cycle the charge stored in the capacitor is positive. The read operation is accomplished by sensing the BL voltage, which depends on charge sharing between C_{bl} and C_{fe}.

Figure 9 reports the bit line voltage waveform showing the successfully writing/reading of data in the generic ferroelectric cell accessed by activating the corresponding WL.

Finally, we implemented expression (3) in Eldo by using HDL-A in order to verify the numerical reliability of the produced code and compare simulation results with respect to the *I.M.A.Ge.* approach.

The HDL-A solution reproduces the shape of the hysteresis loops, but, as clearly shown in figure 10, is

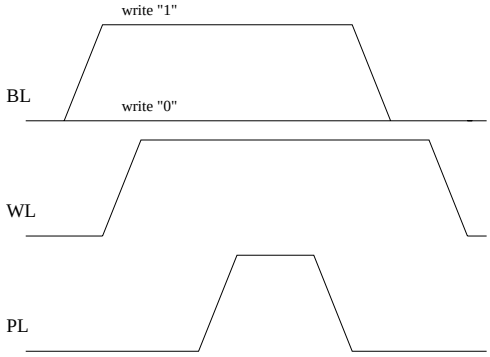

Figure 8: Timing diagram for a typical write cycle operation of the 1T-1C ferroelectric memory cell.

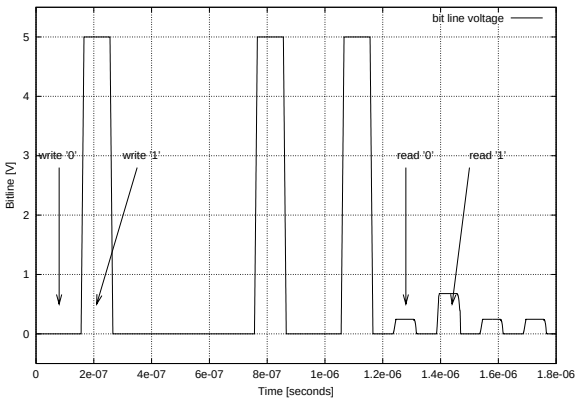

Figure 9: Simulated bit line waveform for a memory write/read cycle.

slower than the numerical algorithm previously presented. In particular, for a number of cells greater than 2000, we obtained a dramatic increase of the CPU time with the HDL-A implementation.

5 Conclusions

The main goal of the research presented in this work was to demonstrate the flexibility and versatility of *I.M.A.Ge.*, a CAD tool developed to embed new semiconductor device models inside circuit simulators.

In particular, we described the implementation of a ferroelectric capacitor model in a commercial circuit simulator.

The proposed ferroelectric model was successfully validated by simulating a complete array of 64k memory cells with WL/PL architecture and showing better performances with respect to other possible behavioral implementation.

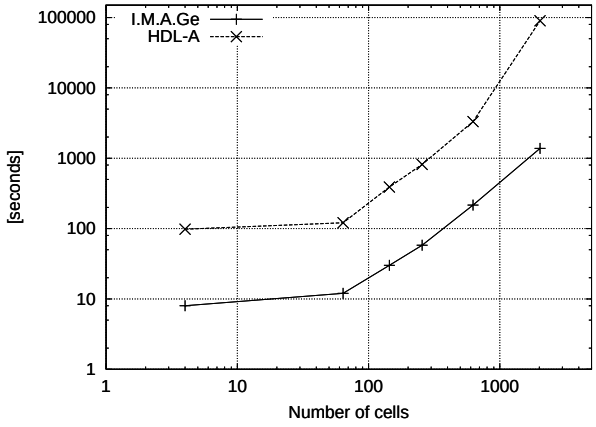

Figure 10: CPU time comparison between HDL-A and *I.M.A.Ge.* ferroelectric memory array implementation.

6 References

[1] *"HDL-A Reference Manual"*, Mentor Graphics, 1995.

[2] *"Eldo UDM User's Manual"*, Mentor Graphics, 1999.

[3] *"SPECTRE HDL Reference"*, Cadence Design Systems, 1998.

[4] V. R. Kasulasrinivas and H. W. Carter, *"Modeling and Simulating Semiconductor Devices using VHDL-AMS"* BMAS 2000

[5] P. J. Ashenden, *"The Designer's Guide to VHDL"*, Morgan Kaufmann Publishers, Inc. 1996.

[6] *"Eldo User's Manual"* Mentor Graphics, 1998.

[7] F. L. Cox, W. B. Kuhn, J. P. Murray and S. D. Tynnor, *"Code-Level Modeling in XSPICE"*, Proc. of ISCAS'92. Vol 2 , 1992 , pp: 871-874.

[8] A. Vladimirescu, *"The SPICE Book"*, John Wiley & Sons, Inc. 1994.

[9] T. Quarles, *"Adding Devices to Spice3"*, Memorandum No. UCB/ERL M89/45 24 April 1989.

[10] S. L. Miller, R. D. Nasby, J. R. Schwank, M. S. Rodgers and P. V. Dressendorfer, *"Device modeling of ferroelectric capacitors"* Journal of Appl. Phys. 68(12), 1990.

[11] A. Sheikholeslami and P. G. Gulak, *"A Survey of Circuit Innovations in Ferroelectric Random-Access Memories"*, Proc. of the IEEE, vol 88, May 2000.